Programação Visual
Trabalho de Laboratório nº 4

Objetivo	MVC – Introdução. Aplicações básicas com definição de regras de encaminhamento e secções.
Programa	Pretende-se criar o *site* privado **IPSCard** de gestão de cartões pré pagos.
Regras	Implementar o código necessário e testar no fim de cada nível. Use as convenções de codificação adotadas para a linguagem C# e para o modelo MVC.

Nível 1	• Abre a solução fornecida com este enunciado (IPSCard).
	• Na pasta Models, crie um novo modelo PrePaidCard, com as seguintes propriedades:
	  ○ `Id (Guid) (com um private set)`
	  ○ `HolderName (string)`
	  ○ `ExpiryDate (DateTime)`
	  ○ `Credit (decimal)`
	• Crie um **construtor sem parâmetros**, na sua implementação:
	  ○ para o `Id`, atribua um novo `Guid` através do método `Guid.NewGuid()`
	  ○ para o `ExpiryDate`, obtenha a data atual (`DateTime.UtcNow`) e acrescente 5 anos (`AddYears`)
	• Crie um **construtor com 2 parâmetros**:
	  ○ `holderName (string)`
	  ○ `startingCredit (decimal)`
	• Neste novo construtor atribua os valores do `HolderName` e `Credit`. Para finalizar, utilize Constructor Chaining para chamar o construtor sem parâmetros.

Nota: Constructor Chaining é uma técnica que permite chamar outros construtores e usa-se a keyword "this" antes do corpo do método:

```
public PrePaidCard(...) : this() {...}
```

Programação Visual
Trabalho de Laboratório nº 4

Nível 2

- Crie uma nova pasta "Data" dentro do projeto e dentro desta, crie uma nova classe estática chamada `SeedCards`.
- Nesta nova classe adicione o seguinte método:

```
public static List<PrePaidCard> Seed() => new ()
{
   new PrePaidCard("Bruno Teixeira", 1000m),
   new PrePaidCard("José Cordeiro", 1500m),
   new PrePaidCard("Vitor Xavier", 2000.50m)
};
```

- Crie um novo controlador **CardsController** (*template* **MVC Controller - Empty**) e defina um campo **privado estático** do tipo `List<PrePaidCard>` chamado cards, e instancie este campo utilizando o método estático `SeedCards.Seed()`.
- No método `Index`, passe o campo `cards` como parâmetro (model) do método `View`.
- Crie uma nova pasta "Cards" dentro da pasta "Views", e dentro desta adicione uma nova View (utilize Scaffold com o template "Razor View"):
    - Nome: Index
    - Template: list
    - Model: PrePaidCard
- Na View do layout (_Layout.cshtml) acrescente um item no menu para o nome controlador e ação.
- **Corra a aplicação, e utilize o menu para ir para a página dos cartões, deverá ver a lista com 3 cartões.**

Programação Visual
Trabalho de Laboratório nº 4

Nível 3

- Crie uma nova View, na pasta "Views/Cards" semelhante ao que fez anteriormente, mas com o name e template "Create".
- Na nova View, apague o código relacionado com o campo `Id` e `ExpiryDate`.
- Adicione a ação Create no controller Cards.

Nota: É suficiente retornar apenas `View()` pois o .NET compara o nome do método aos nomes das Views disponíveis.

Por defeito, o .NET interpreta as ações (métodos que retornam IActionResult) como métodos que correspondem ao verbo HTTP GET. Para podermos indicar que um determinado método utiliza outro verbo, devemos utilizar atributos/anotações (annotations).

- Acrescente uma nova ação "Create", com um parâmetro do tipo PrePaidCard, acrescente este novo cartão à lista de cartões (cards) e retorne a View "Index", passando a lista de cartões como modelo.
- Adicione ainda nesta nova ação o atributo "HttpPost".
- Corra novamente a aplicação, aceda ao ecrã dos cartões e utilize o link para criar um novo cartão. Preencha os campos, e crie o cartão, **verifique que voltou a página da lista dos cartões e consegue ver o novo cartão.**

Nível 4

- Para facilitar ao utilizador encontrar os cartões de uma determinada pessoa, vamos acrescentar um campo de pesquisa na página dos Cartões (View Index), para tal, utilize como exemplo o formulário gerado na View Create, e acrescente um formulário com um campo que irá receber o nome (`<input name="holderName" class="form-control" />`) a pesquisar.
- Para que este novo formulário tenha um método para onde enviar os dados, acrescente uma nova ação (Index) no `CardsController` com o atributo "HttpPost", com um parâmetro `holderName` do tipo `string`. Na sua implementação, utilize LINQ para retornar a lista de cartões filtrada como Model da View (Utilize o método `Contains`).
- Acrescente ainda a lógica necessária para que o sistema mostre todos os cartões quando utiliza a caixa de pesquisa sem escrever nada.
- **Corra a aplicação, e teste o campo de pesquisa.**
- De forma a facilitar ainda mais o acesso a pesquisa, vamos mover o formulário para uma vista parcial para podermos incluir em múltiplas páginas.
- Dentro da pasta "Views/_Shared", crie uma nova View "_CardHolderSearch " (Razer View – Empty) e mova o formulário da View Index para esta nova View parcial.
- Adicione a Partial View à View Index e Create usando a Helper Tag partial (**não vai necessitar do atributo `model` para este caso**).

Programação Visual
Trabalho de Laboratório nº 4

Nível 5

- De forma a podermos alterar o crédito dos cartões vamos utilizar a ação de "Edit". Para tal, na View Index, substitua a linha referente à ActionLink Edit por:

```
@Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
```

- Crie uma nova View, na pasta "Views/Cards", (utilize Scaffold com o template "Razor View"):
  - Nome: Edit
  - Template: Edit
  - Model: PrePaidCard
- Na nova View desative os campos do `Id, HolderName` e `ExpiryDate` com o atributo `disabled` do HTML.
- No CardsController acrescente uma nova ação "Edit" que recebe como parâmetro o `Id (Guid)` e retorna a View Edit passando como modelo o cartão que encontrar através do parâmetro `Id`. Caso não encontre retorne para a View Index.
- Ainda no `CardsController`, acrescente outra nova ação Edit, desta vez com o atributo "HttpPost", que recebe o `Id` mas também o objeto editado do tipo `PrePaidCard`. Neste método procure o objeto na lista e altere o seu `Credit` usando o `Credit` do objeto editado que recebe como parâmetro. Retorne a View Index.
- **Corra a aplicação e experimente trocar o crédito de um dos cartões, verifique que quando volta à lista o crédito ficou alterado.**

Desafio

- Implemente as restantes ações CRUD para apagar um cartão e para visualizar os detalhes de um cartão. Pode usar os *templates* respetivos na criação das vistas associadas.

Notas

Para os identificadores siga as convenções adotadas pelo C#, nomeadamente:

- A notação camelCase para o nome das variáveis locais e identificadores privados.
- A notação PascalCase para os nomes públicos dos métodos e classes
- Não utilize o símbolo '_' nos identificadores
- Não use abreviaturas